

周报

本周主要完成两方面的工作

1. 教材修改初稿完成
2. SimHash 开始代码阅读、重构和压缩域体绘制框架搭建

教材的修改，本周主要加上了 obb 树碰撞检测的算法伪代码，如下：

传统的基于 OBB 树的碰撞检测算法一般可分为两个阶段，一是预处理阶段，另一个是碰撞检测阶段。预处理阶段就是把物体进行分解，为分解得到的每个层次的物体块构造相应的 OBB 包围盒。碰撞检测运行阶段，遍历物体对的层次二叉树，层次树中的每一个节点与 OBB 包围的分解块相对应。遍历层次树时，先检测分解块的节点所对应的 OBB 包围盒是否相交；当两 OBB 包围盒相交时，对两 OBB 包围盒进行三角形的相交测试，得出碰撞检测结果。在进行 OBB 碰撞检测时，又分为两步，OBB 间的重叠测试和基本几何元素之间的相交测试。使用 OBB 树进行碰撞检测，其目的是通过两个对象的包围盒树中各节点所对应的 OBB 包围盒之间的重叠测试，尽可能早地排除不可能相交的基本几何元素对，仅对有可能相交的基本几何元素对进行精确的相交检测。OBB 包围盒间的重叠测试即可通过分离轴定理计算，而基本几何元素的相交测试则可通过 6.4.2 中介绍的求交算法计算。例程 6-11 给出了一个在游戏开发实例中使用 OBB 树进行碰撞检测的伪代码，它本质上是对图 6-12 流程图的展开和实现。

例程 6-11 基于场景 OBB 树的碰撞检测实例应用伪代码

```
InitRenderer(){//初始化阶段
    LoadTextures();//载入纹理
    LoadScene(Filename);//载入游戏场景和对象
    CreateVBOIBO();//创建顶点缓冲区对象和索引缓冲区对象
    InitOBBDTree(numFaces, numVertices, Filename);//使用模型网格和参数，初始化各自的 OBB 树
    BuildOBBDTree();//为场景中所有对象构建各自 OBB 树和 AABB 包围盒
    CoarseCollisionDetection();//使用 AABB 包围盒进行初步的碰撞检测，找到有可能发生碰撞的对象
}
//使用分离轴算法，测试两个待检测对象的 OBBDTree 是否发生碰撞
OBBDTreeCollision(g_object1, g_object2){
    以 g_object1 和 g_object2 的 OBBDTree 根节点构造第一对待检测包围盒对，压入栈 obbox_pairs
    while(obbox_pairs不为空){
        curNodeA和curNodeB是待检测的包围盒；
        //若当前OBB包围盒发生碰撞，则继续进入下一层进行检测
        if(OBBDBoxPairsCollision(curNodeA, curNodeB)){
            if(curNodeA和curNodeB是叶子节点){
                设置 curNodeA和curNodeB为最终发生碰撞的包围盒
            }
            else if (只有curNodeA是叶子节点){
                分别以curNodeA和curNodeB的两个子节点作为碰撞对，入栈obbox_pairs;
            }
            else if (只有curNodeB是叶子节点){
                分别以curNodeB和curNodeA的两个子节点作为碰撞对，入栈obbox_pairs
            }
            else{// curNodeA和curNodeB都不是叶子节点
                if(curNodeA的包围盒面积大于curNodeB的包围盒面积){
                    分别以curNodeB和curNodeA的两个子节点作为碰撞对，入栈obbox_pairs;
```

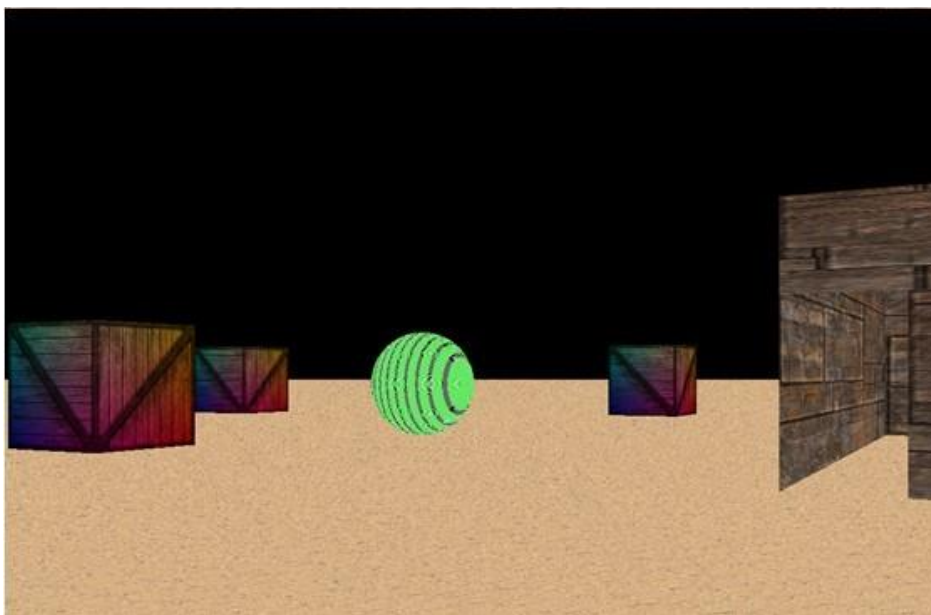



图 6-13 基于 OBB 树碰撞检测的三维游戏场景

此外，在完成教材的修改后，开始 SimHash 压缩项目的研究和实现。首先，下载并阅读了不同的几个 SimHash 代码实现，包括 GOOGLE 那篇文章作者的实例代码，最终还是决定自己重新写一个。因为这些代码或者是基于 java 或者 python，或者就是已经紧耦合的嵌入进数据库实例应用，或者是上述情况的综合，而且作者为了追求代码的效率，牺牲了可读性，看起来相当费力。算法的核心基本已经弄得很明白，目前正在实现中。总体框架，基于 QGLviewer3D 图形库（主要用于相机控制，绘制场景控制和交互等），vroeen 的传输函数界面以及伟峰之前的体绘制代码，重构了一个基本框架。SimHash 正在实现中，如果顺利，下周会有初步结果。目前预测的一个困难是，SimHash 直接应用与整形数据的操作中比较顺乎自然，而当用于浮点数据的处理时，可能会遇到一些麻烦，目前还不清楚会有什么样的问题。先实现基本的看看。